Applications in Practical High-End Computing - Group Project

Assignment - "Workflow"

Design Document

Supervisor: Dr Stuart Barnes

Authors: Mateusz Gołąb
Csaba Kerti
Jakub Kiełbasa
Zsolt Kollarits

Course: CSTE / SETC

Table of contents

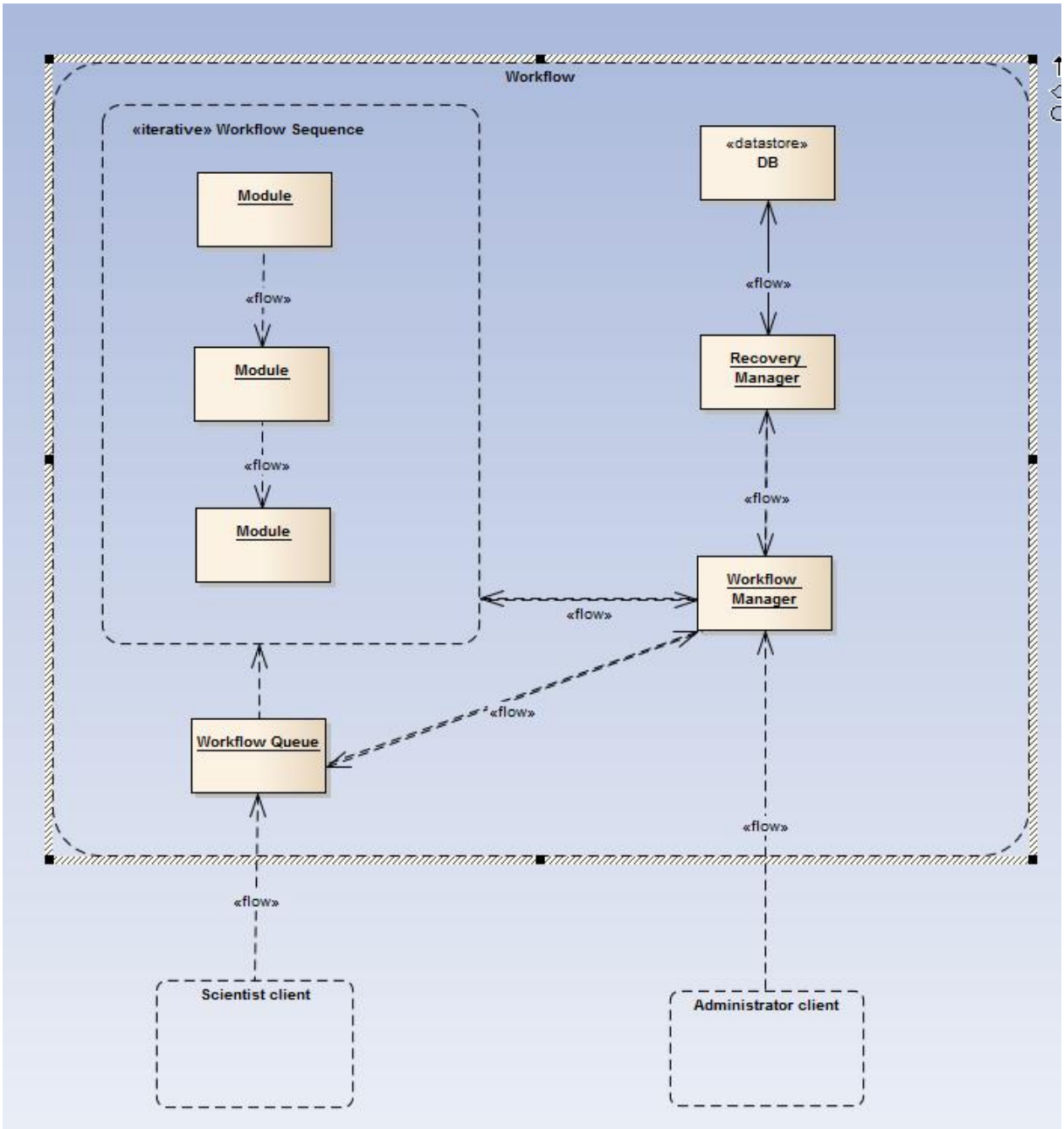
1. Introduction

This document contains detailed design information of Workflow system. Includes design patterns, architectural models, Workflow configuration and recovery mechanism description.

2. Architecture

System is designed according to client – server architecture. Server part comprises of Workflow and there are two types of clients , Scientist client and Administrator client, which both have access to Workflow via terminal.

2.1 Workflow outline



Workflow outline

2.2 Administrator's relationship with the system

Administrator has a possibility to create new or modify existing workflows. These functionalities are reachable through the administrator client program.

The program allows to the administrator to remove or add arbitrary number of modules to the actual workflow. The administrator should set an XML descriptor file to each module, which describe how to execute the module. Furthermore the administrator should set two XSD validator files to ensure the input and output validation before and after each module.

Administrator also needs to set an XML descriptor for the structure of starting parameters, which helps to the scientist's client application to send scientist's simulation parameters to the system.

After the administrator completes the edition on the workflow and save it, the system is ready to use for the scientist.

2.2 Scientist's relationship with the system

The scientist can reach the system through a client program. This application helps them to enter the parameters for the simulation. After the parameters have been entered, the system processes them and creates a unified XML file which contains the parameters. The simulation probably won't start immediately, the XML file will be added to the WorkflowQueue. The system continuously processes the simulations from this queue, so sooner or later the simulation will be started.

The scientist can use the same program to monitor its simulation and to check runtime logs.

The result of the simulation will be in the directory of the scientist when the simulation will be finished.

3. Design Patterns

When designing workflow application we can consider group of special design patterns specified for such type of programs. According to Van der Aalst classification such group comprises of control flow patterns, data patterns , resource patterns, error handling patterns and presentation patterns. On workflow component level we can consider using universal design patterns for software development. [1]

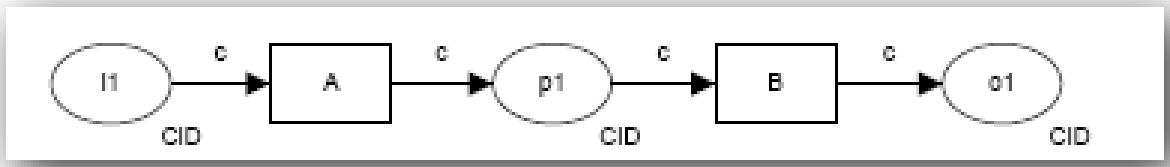
3.1 Workflow specific design patterns

This chapter contains some Workflow type specific design patterns according to W.M.P van der Aalst classification.

3.1.1 Control flow pattern : Sequential workflow pattern

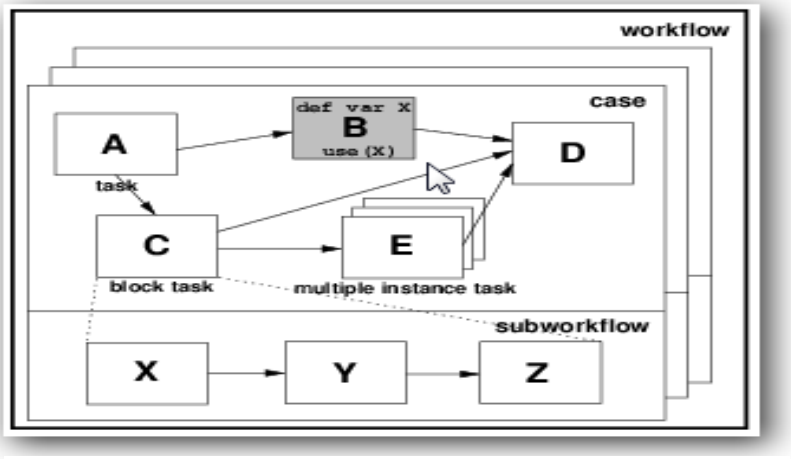
The idea of this pattern is to enable a task in a process after the completion of a preceding task in the same process. The workflow controls the sequence of activities and decides which of the steps will execute next. Such pattern is appropriate for developing workflows if you must execute a series of predefined steps to accomplish a certain task. The sequence pattern is used to model consecutive steps in a workflow process and is directly supported by each of the workflow management systems available. The typical implementation involves linking two activities with an unconditional control flow arrow.

The next figure shows the structure of this pattern, where A and B are the *modules* and l1, p1, o1 are *the format validation parts*. [2]



3.1.2 Data pattern : Task Data

Main idea of this pattern is that data elements can be defined by tasks. Every task is accessible only within the context of his individual execution instance. This pattern suits perfectly requirements of workflow we are developing. According to requirements every module in workflow has his own data to operate on. In our case there is no need to create and manage shared data space for every module placed in workflow . Every executed module operate on his own data. [3]



Gray color represents data ,which is accessible only to B module.

3.1.3 Resource pattern : Automatic execution

The most important property of this pattern is the ability for an instance of a task to execute without needing to utilise the services of a resource. The mechanism of the

program's resource handling can be implemented using this pattern**.** Where a task is nominated as automatic, it is initiated immediately when enabled. In other words, program executed as a module within workflow manages all available resources provided by system. [4]

3.1.4 Exception handling pattern : Recovery action

When specifying a recovery action , the point in the process which should be chosen to restart process is last successfully finished work item by default. Although it can be any preceding point in the process. The effectiveness of recovery strategy is largely governed by the richness of the events captured in the execution log and some events (e.g. resource allocation, production of hard copy reports). [5]

Exception types :

1. **Work item failure**

Failure of single work item during the execution of a workflow process is generally characterised by the incorrect output provided by this item. It can be caused by software or hardware error. Failure of some network component associated with work item can cause some errors as well. Especially when module is working on remote server.

2. **Deadline expiry**

It is common to specify a deadline for a work item in a workflow process model. Usually it indicates when the work item should be completed although timeouts for commencement are also possible. In general with a deadline , it is also useful to specify at design time what should be done if the deadline is reached and the work item has not been completed.

3. **Resource Unavailability**

It is often the case that work item requires access to one or more data resources during its execution. If these are not available to the work item , then it is usually not possible for the work item to finish successfully. Although the occurrence of these issues can be detected, quite often cannot be resolved within the context of the executing process.

3.2. Application specific design patterns

There are two design patterns used in terms of application level design patterns. Model View Controller and Observer.

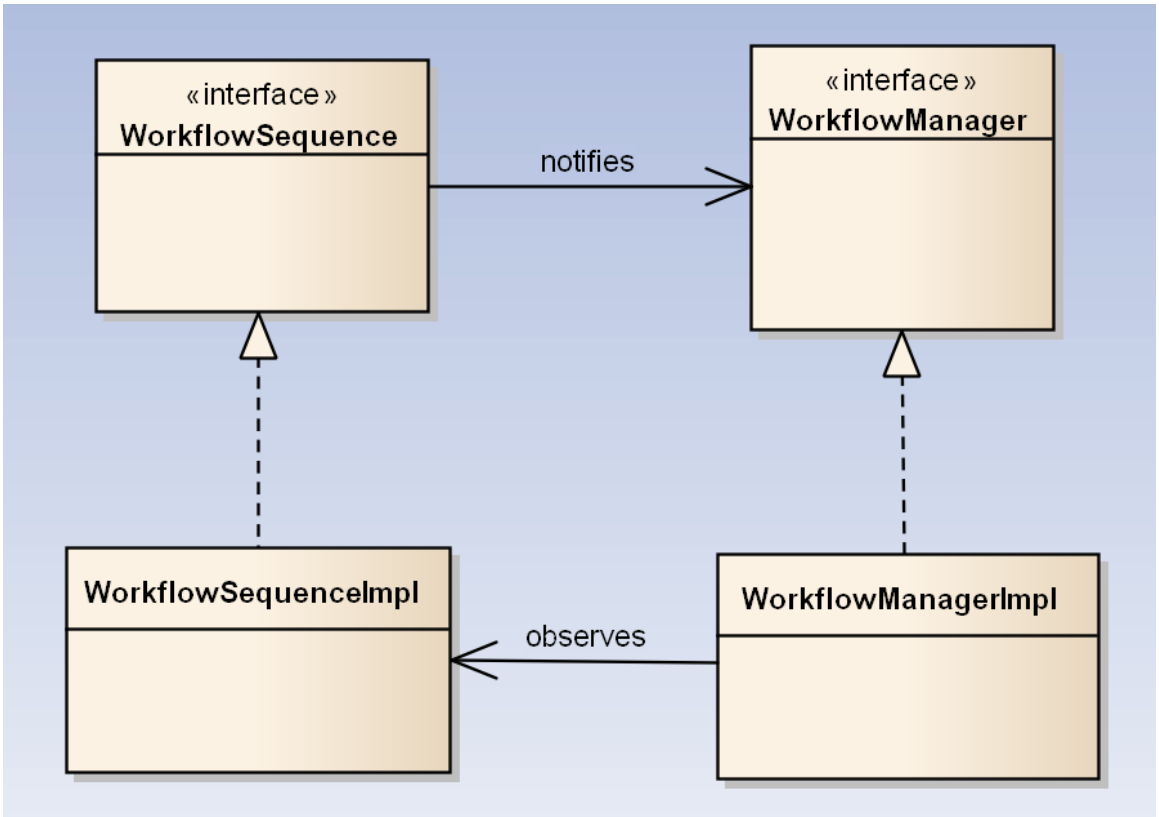
3.2.1 Model View Controller

All responsibilities in program are divided into separated aspects , according to MVC pattern. Model comprises of *Simulation, StablePoint, ModuleParameter* and *Module* clases. *WorkflowManager, WorkflowSequence, WorkflowQueue, DatabaseManager* and *ModuleDataValidator* are controller's classes. View classes are responsible for results visualisation and graphical interface. Main view classes are : *WorkflowManagerView* and

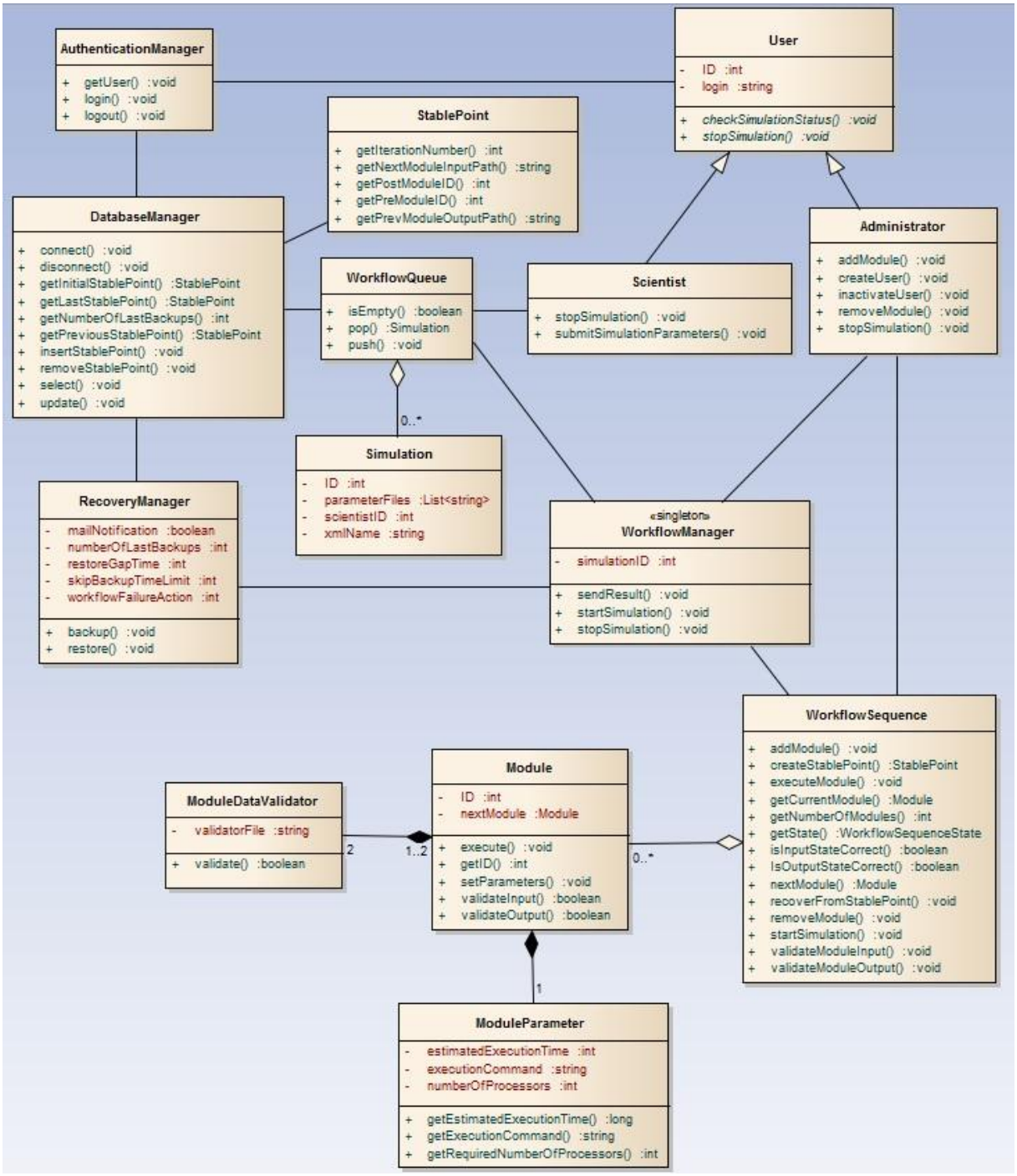
WorkflowSequenceView. MVC pattern isolates application logic from the presentation by separation of aspects, providing a loose coupling between model , view and controller.

3.2.2 Observer

Another software design pattern used in Workflow is the observer. The main idea of this pattern is to provide dependency mechanism between objects , so that when one object changes state, all its dependents are notified. *WorkflowSequence* is the subject (observable) which notifies observer about changes and activities in current simulation. *WorkflowManager* is the observer which reacts on *WorkflowSequence* notifications.



4.Class diagram



1. Authentication Manager

This class represents authentication mechanism . Performs user log in, log out operations . This manager is also responsible for adding and inactivating users.

2. Workflow Manager

Main controller which manages all Workflow components. Responsible for running simulations as well as validating and monitoring each module. Decides when to use Recovery Manager.

3. Recovery Manager

The Recovery manager implements one of the most important feature of the Workflow – recovery mechanism. This component manages backup and restore mechanisms and keeps logic of Workflow recovery policy.

4. Database Manager

This component is directly connected to system database. Provides high level methods for other Workflow components to get or store specific data.

5. Workflow Sequence

This component represents sequence of modules present in actual simulation.

6. Module

Component which represents executable program/script, which is running in a workflow. Administrator provides every module to the Workflow together with :

- ✓ Module metadata files (commands/parameters XML)
- ✓ Format validation files (XSD)

7. Module Data Validator

This class performs validation of input or output of specified Module. Operates on files containing validation rules.

8. Module Parameter

Represent set of module parameters , provided in xml file. These parameters are essential for module execution and monitoring.

9. Workflow Queue

This component logically represents queue containing necessary parameters for module execution , provided by Scientist.

10. Simulation

Represents simulation which runs in Workflow Sequence or waits in Workflow Queue. Contains parameters provided by Scientist which are essential to start simulation. Every simulation is unique within the Workflow and belongs to some Scientist.

11. User

The class which represents user actor

12. Scientist

The class which represents Scientist actor

13. Administrator

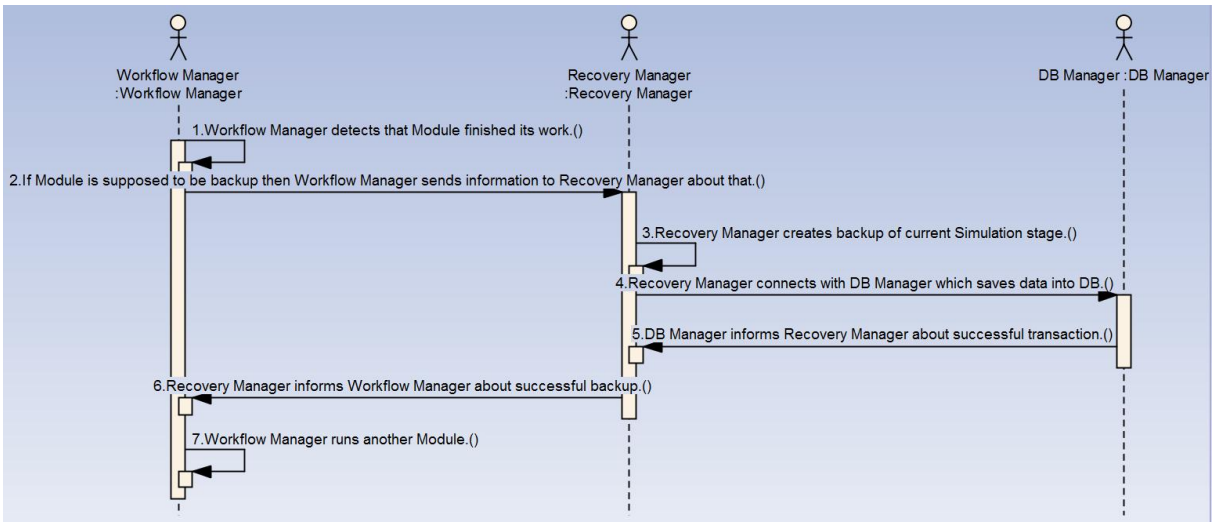
The class which represents Administrator actor.

5. Recovery mechanism

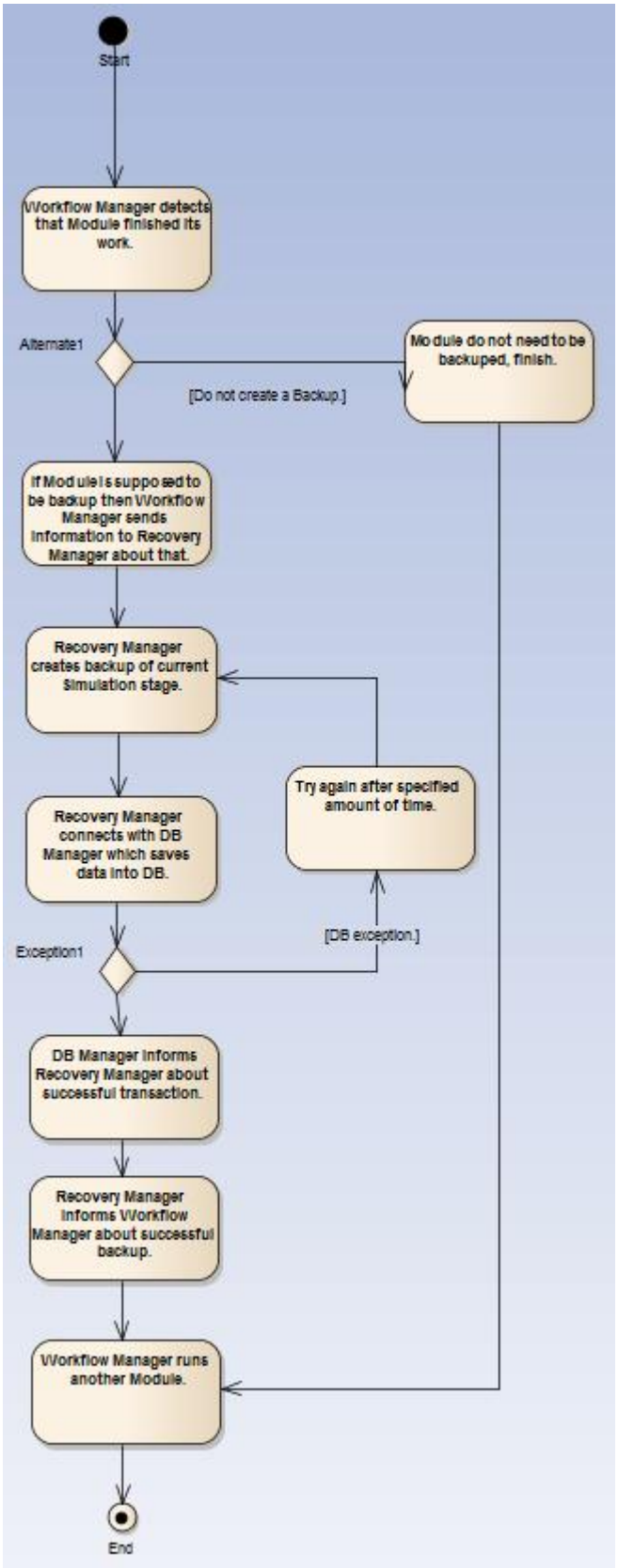
5.1.Backup policy

Performing a backup of stable point in Workflow Sequence is essential activity to enable recovery mechanism. Decision when backup should be done is made dynamically. Execution time of every module is measured and compared with value of configurable parameter. If module's execution time is greater than parameter's value, backup of Workflow Sequence state will be performed. It is possible that every module's execution time will be less than specified in parameter. Therefore after every iteration, Workflow manager checks how many backups were made in iteration. If none, Recovery manager will perform backup. Such mechanism ensures at least one backup per iteration and allows dynamic backup policy.

5.1.1 Sequence diagrams



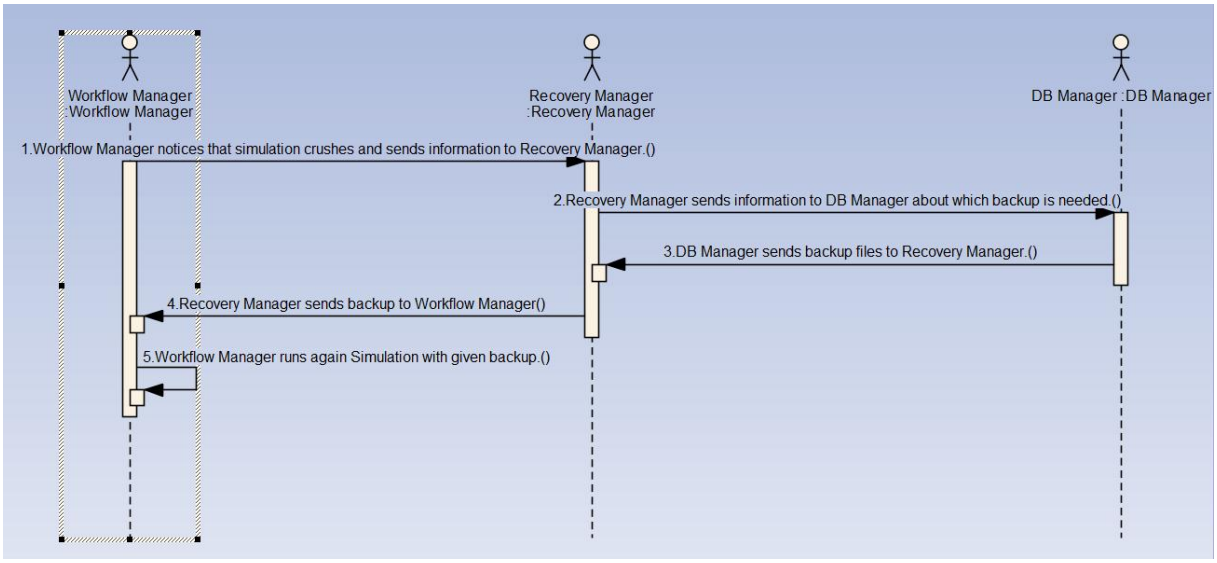
5.1.2 Activity diagram



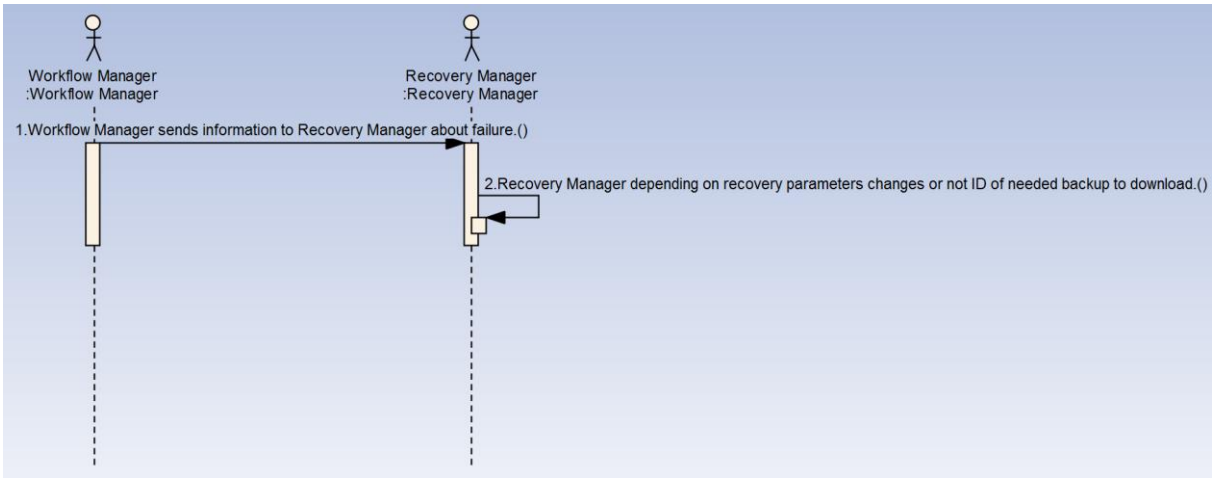
5.2. Recovery

The main aim of recovery mechanism of the Workflow system is to react on failure caused by module, by restarting workflow from some previous stable point in Workflow Sequence. When error occurs in Workflow Sequence , Workflow Manager starts recovery mechanism by launching Recovery Manager . First activity of this component is receiving last stable point of Workflow Sequence from Database Manger and sending it to Workflow Manager to restart Workflow Sequence. When error occurs after these activities , Recovery Manager waits recovery gap time which is specified by Administrator in configuration file and then again provides last stable point to Workflow Manager. This gap between recoveries gives possibility to wait if there is some problem with remote server which hosts a module, instead of performing another recovery instantly. This mechanism is configurable , so Administrator can decide whether it is useful for actual workflow. When failure occurs again, previous stable point is provided after recovery gap. Scenario of using previous stable point in case of error is repeated. Total number of recoveries is limited by parameter in configuration file. If workflow failed despite recovery attempts one of the actions specified by parameter is undertaken. Possible actions are : restarting whole simulation, terminating simulation or performing recovery process again.

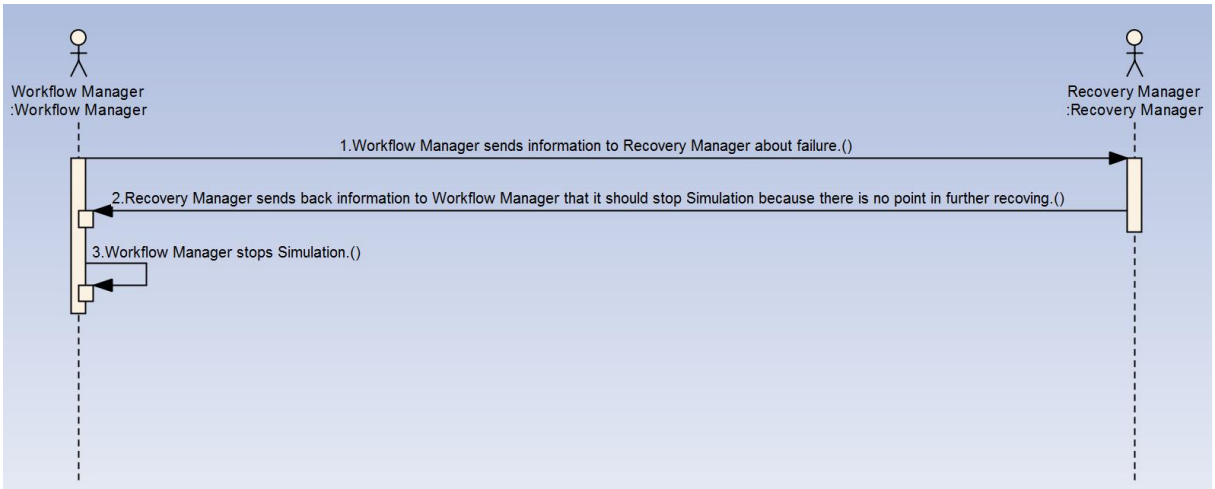
5.2.1 Sequence diagrams



Basic Path

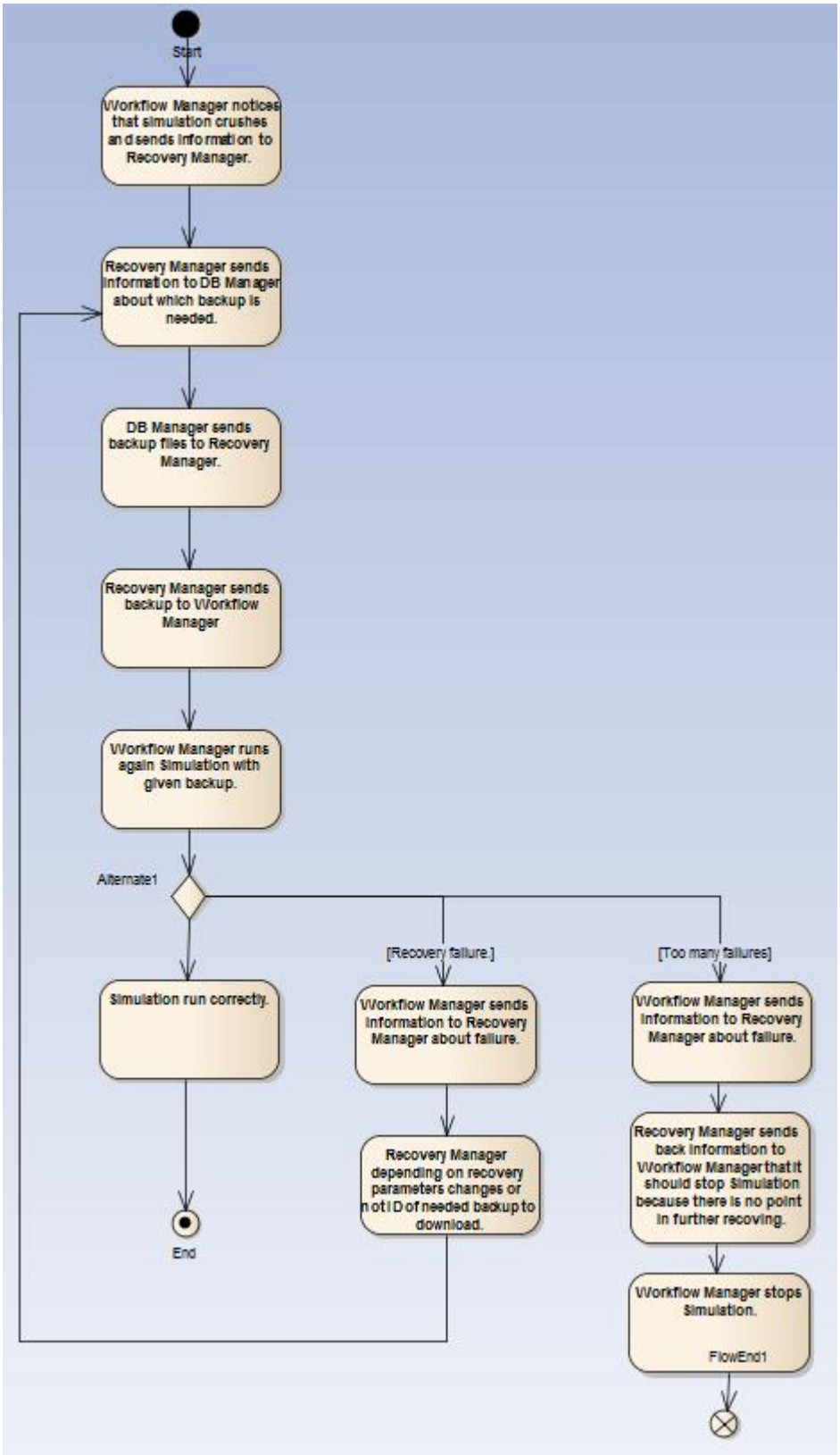


Failure after recovery



Too many failures after recoveries

5.2.2 Activity diagram



6. System configuration

Administrator has possibility of configuring some Workflow and Recovery mechanisms using configuration files.

6.1 conf.xml

Contains Recovery parameters.

Content:

```
<skipBackupTimeLimit value=10000>
```

This parameter describes time in seconds which is compared to execution time of every module . If execution time is less than parameter, backup will be skipped .

```
<numberOfLastBackups value=3>
```

Represents number of stable points recovery attempts within one recovery process.

```
<restoreGapTime value=500>
```

Describes time in seconds which elapses between every single recovery attempt.

```
<workflowFailureAction value=(RESTART_SIMULATION|TERMINATE|RESTART_RECOVERY)>
```

This parameter represents action which is undertaken after unsuccessful recovery process.

- RESTART_SIMULATION - simulation is restarted.
- TERMINATE – simulation is terminated, next simulation from the Worker queue starts.
- RESTART_RECOVERY – recovery process is restarted.

```
<mailNotification value=(TRUE|FALSE)>
```

This parameter controlls email notification mechanism. Notification can be successfully finished simulation or workflowFailureAction value.

6.2 moduleParameters.xml

Contains bunch of information for each module.

Content :

```
<estimatedTime value=10000>
```

Describes estimated execution time of module.

```
 <numberOfProcessors value=3>
```

This parameter represents number of processors required to launch specified module.

```
<executionCommand value="ssh 135.123.456.789 ./module1 -50 mesh.dat">
```

This parameter contains full module's execution command. If module is installed on remote server, execution command has to contain connection command.

6.3 scientistParameters.xml

This file contains parameters provided by Scientist to run new simulation in Workflow. Parameter included in this file can be set of values of any type or link to the file containing big set of values.

6.4 initialParametersSkeleton.xml

Represents structure of input parameters for the first module in the Workflow Sequence. Contains types of every parameter in specified order. This file is provided by Administrator.

6.5 validator.xsd

This file is responsible for module input/output validation. Contains the rules which ensure correctness of validated file.

7.References

[1] *Workflow Patterns.*

W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros.
Distributed and Parallel Databases, 14(3), pages 5-51, July 2003

[2] *Workflow Control-Flow Patterns*: *A Revised View*.

N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar.
BPM Center Report BPM-06-22 , BPMcenter.org, 2006.

[3] Workflow Data Patterns**.**

N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst.
QUT Technical report, FIT-TR-2004-01, Queensland University of Technology, Brisbane, 2004.

[4] Workflow Resource Patterns**.**

N. Russell, A.H.M. ter Hofstede, D. Edmond, and W.M.P. van der Aalst.
BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven, 2004.

[5] *Exception Handling Patterns in Process-Aware Information Systems.*

N. Russell, W.M.P. van der Aalst, and A.H.M. ter Hofstede.
BPM Center Report BPM-06-04 , BPMcenter.org, 2006